UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório do Projeto Orientado em Computação II - POC II

Adequação das Funcionalidades de Indexação de Documentos no HyperPro Versão 1.1

Orientanda: Luciana Leal Ambrosio
Orientadora: Mariza Andrade da Silva Bigonha

28 de junho de 2001

Sumário

Resumo

O HyperPro é um ambiente de desenvolvimento de programação em lógica baseado no paradigma de estilo literário, que foi especificamente desenvolvido para a linguagem de programação Prolog. Seu objetivo é possibilitar ao usuário combinar em um mesmo documento a escrita e a documentação de programas CLP. O protótipo do HyperPro já está sendo usado desde o final de 1999, porém durante este ano de uso alguns problemas foram detectados em algumas funcionalidades, especificamente a paginação, que não estava funcionando, impedindo que o índice de referência cruzada funcionasse corretamente. A paginação foi então implementada e já se encontra disponível para o HyperPro Versão 1.1. O índice de referência cruzada continuou não funcionando para todos os casos. Parte do seu erro foi identificado, porém ainda não foi consertado, sendo válido como trabalho futuro. Dessa forma, o sistema HyperPro Versão 1.1 está mais robusto, tendo funcionalidades importantes que ainda não haviam sido implementadas.

Palavras-Chaves:

HyperPro, paradigma de estilo literário, paginação de documentos, índice de referência cruzada

1 Introdução

O sistema HyperPro [1] é um ambiente de desenvolvimento de programação em lógica baseado no paradigma de estilo literário [7], especificamente desenvolvido para a linguagem de programação Prolog. Seu objetivo é possibilitar ao usuário combinar em um mesmo documento a escrita e a documentação de programas CLP *(Constraint Logic Programming)* [8]. Para tal, ele oferece ao usuário a possibilidade de editar, em um ambiente homogêneo e integrado, diferentes versões de programas Prolog, comentários e informações de verificação formal, assim como a possibilidade de executar, depurar e testar os programas [13].

CLP é uma área de pesquisa que pode ser localizada entre Inteligência Artificial e Linguagens de Programação e diz respeito a modelagem, solução e programação de problemas da vida real. Estes podem ser descritos como um conjunto de afirmações (constraints), que impõem relações entre as variáveis do problema. CLP descende de programação em lógica. Prolog é uma linguagem baseada em Lógica de Primeira Ordem, que define relações [2].

O HyperPro faz uso da ferramenta Thot, que é um sistema para produzir documentos estruturados [10]. Esta ferramenta permite ao usuário criar, modificar e consultar interativamente documentos que obedecem modelos, permitindo a produção de documentos homogêneos. Thot ainda possui várias facilidades como hypertexto, atualização de referência cruzada e construção de tabela de índices. Como a formatação do documento é feita pelo sistema, isto possibilita ao usuário focalizar na organização e no conteúdo dos documentos.

Um documento HyperPro é basicamente um documento Thot. Ele deve conter um título, uma sequência de pelo menos uma seção, uma tabela de conteúdo e um índice de referência cruzada. Data, nomes dos autores e suas afiliações, palavras-chave, referências bibliográficas, anexos e um índice de versões são informações opcionais que podem ser gravadas [1].

O sistema HyperPro provê as seguintes funcionalidades: diferentes visões estáticas de um documento, testes de diferentes versões de um programa, indexação, visões dinâmicas de um documento, verificação sintática para diferentes linguagens CLP e exportação de documentos.

Este trabalho concentra-se nas funcionalidades de paginação de documentos e de indexação, especialmente a de índice de referência cruzada. Estas funcionalidades já foram implementadas no protótipo [14]. Nosso objetivo é torná-las mais robustas na versão 1/2001. A indexação é um mecanismo, que utilizando-se da funcionalidade de paginação, encontra referências e ocorrências de uma relação em páginas de um documento, como por exemplo: a localização da definição de um predicado bem como a localização de seu uso em outras versões do programa ou do documento. A indexação de um documento possui várias vantagens. Uma delas é que, por meio da projeção baseada na indexação, partes do documento original podem ser visualizadas selecionando-se a informação apropriada diretamente a partir do índice.

Na seção 1 deste documento se encontra a motivação deste trabalho. A seção 2 dá uma visão mais aprofundada do sistema Thot, já a 3 dá uma visão mais aprofundada do sistema HypePro. A seção 5 explica o trabalho desenvolvido, a 6 mostra exemplos das funcionalidades implementadas e a 7 dá uma conclusão do trabalho. O apêndice A contém o manual do usário para as funiconalidades implementadas.

2 Motivação

O HyperPro faz uso do paradigma da programação literária, um paradigma muito importante para a compreensão de programas e textos. Esse paradigma estabelece que a documentação

e o código de um programa devem estar em um mesmo documento. As ferramentas implementadas para usar esse paradigma permitem que o usuário digite as partes de um programa em qualquer ordem e extraia a documentação e o código do mesmo arquivo, quer seja para gerar um artigo ou para executar um programa e obter um resultado [13].

O protótipo do HyperPro já está sendo usado desde o final de 1999, ocasião de seu lançamento. Durante este ano de uso alguns problemas foram detectados em algumas funcionalidades, especificamente, a paginação não estava funcionando até agora, o que impedia que o índice de referências cruzadas funcionasse corretamente para todos os casos de teste. A remoção desses "bugs", objetivo deste projeto, já foi feita e portanto esta remoção já pode garantir a perfeita execução do sistema HyperPro, além de torná-lo mais robusto.

3 Thot

Thot [10] é um sistema desenvolvido para a produção de documentos estruturados. É um sistema que permite ao usuário criar, modificar e consultar interativamente documentos. Os modelos permitem a produção de documentos homogêneos. O sistema cuida de sua formatação, deixando o usuário se concentrar na organização e no conteúdo. Thot também possui outras operações, tais como numeração, referências cruzadas, índices, além de prover diferentes vistas de um mesmo documento.

Thot é um sistema integrado e extensível. Ele permite o processamento não somente de textos estruturados dentro da mesma ferramenta e dentro de um mesmo documento, mas também de gráficos, tabelas, fórmulas matemáticas, etc. Isto faz com que o Thot não processe apenas uma lista exaustiva de tipos de documentos: o usuário pode adicionar outros tipos de informação, especificando o modelo apropriado do documento.

Todos os serviços que o Thot provê são baseados no sistema interno de representação de documentos [9]. O modelo de documentos do Thot permite que o usuário trabalhe com certas entidades que ele tem em mente quando faz um documento. E são justamente essas entidades lógicas, tais como títulos, capítulos, seções, parágrafos, notas, figuras e referências cruzadas que fornecem ao documento sua estrutura lógica.

Devido a este modelo, o autor pode dividir o documento em capítulos, dando um título a cada um. O conteúdo desse capítulo pode ser dividido em seções, subseções, etc. E esse texto é organizado em parágrafos sucessivos, de acordo com o conteúdo. O modelo de documento do Thot é então baseado no aspecto lógico de um documento. A criação de um modelo essencialmente requer a definição de todas as entidades que podem aparecer nos documentos e das relações entre essas entidades e as formas como elas estão ligadas.

Aparentemente é impossível construir uma lista exaustiva de todas as entidades que são necessárias e suficientes para descrever um documento, assim como também não é possível especificar todos os possíveis arranjos dessas entidades no documento. Então o sistema Thot usa um meta-modelo que permite a descrição de numerosos modelos onde cada um descreve uma classe de documentos que, por sua vez, possuem estruturas muito parecidas. Por exemplo, a coleção de artigos publicados por um laboratório constitui uma classe. Evidentemente não é possível enumerar todas as possíveis classes de documentos, e também é possível que novas classes de documentos sejam criadas para satisfazer novas necessidades e aplicações.

Cada documento possui uma estrutura específica que organiza suas partes. Já a estrutura genérica define as formas pelas quais uma estrutura específica pode ser construída, ou seja, descreve a organização lógica de um documento.

Há uma correspondência de um para um entre uma classe e uma estrutura genérica: todos os documentos de uma classe são construídos de acordo com uma mesma estrutura genérica. Uma classe é, portanto, um conjunto de documentos cuja estrutura específica é formada de acordo com a mesma estrutura genérica. Logo, uma classe é caracterizada por sua estrutura genérica.

Estruturas genéricas apenas descrevem a organização lógica de um documento, não sua apresentação física na tela ou em uma folha de papel. Mas para um documento ser mostrado ou impresso sua representação gráfica deve ser considerada. Os detalhes de apresentação de um documento essencialmente servem para mostrar sua estrutura lógica. A apresentação é dependente da organização lógica do documento.

Da mesma forma que uma única estrutura lógica genérica não pode ser definida para todos as classes de documentos, regras universais de apresentação que se aplicam a todas as classes de documentos não podem ser definidas. Por exemplo, para certos tipos de documentos os títulos dos capítulos são centrados na página e são impressos em letras grandes e em negrito, enquanto que para outros tipos de documentos esses mesmos títulos de capítulos são alinhados na margem esquerda da página e são escritos em letras pequenas e em itálico.

A especificação da apresentação de um documento deve ser baseada na sua classe. Ela deve ser expressada em termos de suas entidades, que por sua vez são definidas na estrutura lógica genérica da classe. O conjunto de regras que especificam a apresentação de todos os elementos definidos na estrutura lógica genérica de um documento é chamado de apresentação genérica.

O fato de haver uma apresentação conectada a uma estrutura genérica e dela ser descrita por uma apresentação genérica acarreta diversas vantagens. A primeira delas é que uma apresentação homogênea para diferentes documentos de uma mesma classe é assegurada aplicando a mesma apresentação genérica para todos os documentos da classe. Outra vantagem é que a apresentação facilita mudanças no aspecto gráfico dos documentos.

As definições de meta-linguagem e de classes de documentos também podem ser aplicadas a objetos. Objetos possuem o mesmo nível de representação lógica dos documentos, contudo, com algumas vantagens. Em particular, é possível definir a apresentação separadas dos objetos e atá-los a classes. Então, como documentos, os objetos do mesmo tipo possuem uma apresentação uniforme e a apresentação de cada objeto em uma dada classe pode ser mudada simplesmente substituindo a apresentação genérica de classe. Uma outra vantagem de se usar esse modeo de documento é que o sistema é transparente ao usuário, permitindo que ele se concentre no aspecto lógico do documento e dos objetos [13].

Para programar no Thot existe disponível no sistema quatro linguagens: *S*, que é responsável pelo Esquema de Estrutura; *P*, responsável pelo Esquema de Apresentação; *T*, pelo Esquema de Tradução e *A* pelo Esquema de Aplicação do documento. Essas linguagens serão descritas nas próximas seções.

3.1 Linguagem S

Estruturas genéricas [9] formam a base do modelo de documento do Thot. Cada estrutura genérica, que define as classes de documentos e objetos, é especificada por um "programa"escrito em uma linguagem S [9], e chamado *Esquema de Estrutura*. Esses esquemas de estrutura são compilados em tabelas, chamadas tabelas de estrutura e são usadas pelo Thot, determinando seu comportamento. A gramática de S, assim como as gramáticas das linguagens P e T são descritas usando a meta-linguagem M, derivada de Bakus Naur Form

(BNF).

Cada esquema de estrutura começa com a palavra-chave *STRUCTURE* e termina com a palavra-chave *END*. A palavra-chave *STRUCTURE* é seguida pela palavra-chave *EXTENSION* no caso em que o esquema define uma extensão, seguida pelo nome da estrutura genérica que o esquema define e de um ponto e vírgula.

Em um esquema completo, a definição do nome da estrutura é seguido pelas declarações de *default*, atribuições globais, parâmetros, regras de estrutura, elementos associados, unidades, elementos de esqueleto e exceções.

As noções de atributos, construtores e elementos estruturados são usados na definição da estrutura lógica e genérica de documentos e objetos.

Um construtor está no nível da meta-estrutura, portanto, ele não descrfeve a existência de relações entre estruturas dadas, mas define como os elementos são passados para a linguagem de montagem para construir uma estrutura conforme o modelo.

O modelo é suficientemente flexível para levar em consideração todas as fases da vida útil de um documento. Por exemplo, se uma estrutura genérica específica deve conter um título, uma citação, uma introdução e pelo menos dois capítulos e uma conclusão, isso significa que o documento deverá conter todos esses elementos. Quando o autor começa a escrever, nenhum desses elementos está presente. O editor usa esse modelo e ainda tolera documenos que não estão exatamente iguais a essa estrutura genérica.

O trecho abaixo é um trecho do Esquema de Estrutura que define o protótipo do HyperPro utilizando a linguagem S:

```
{------------------------------------------------------------------------ }
{ Document model for Logic programming environment HyperPro               }
{ Pierre Deransart / Ali Ed-Dbali / Khalid El Qorchi/Flavia Peligrinelli }
{ Fabricio Schmidt                                                        }
{ May, 04th, 1999.                                                        }
{------------------------------------------------------------------------ }
STRUCTURE HyperPro;
DEFPRES HyperProP;
ATTR
    Manual_proj_visible          = Yes;
    Recursive_proj_visible       = Yes;
    Reg_expression_proj_visible  = Yes;
    Index_IV_proj_visible  = Yes;
    Index_CRI_proj_visible = Yes;
    Version_proj_visible         = Yes;
STRUCT
   HyperPro (ATTR First_page_number = Integer;
                  First_section_number = Integer) =
        BEGIN
            Document_title  = Lines;
            CRIndex         = CRI;   {*CRI*}
            IVIndex         = IV;    {*IV*}
          ? Document_date   = Lines;
          ? Authors         = LIST OF (Author);
          ? Affiliations    = LIST OF (Affiliation = Lines);
          ? Key_words    = Lines;
            Sections_seq;
          ? Bibliography   = LIST OF (Citation_biblio = RefBib);
```

```
        ? Annexes         = LIST OF (Annexe);
      END;
   Author (ATTR Author_type = Principal_author,Secondary_author) =
           Content + (Affiliation_ref)
                   - (Rel_def_ref, Current_pred, Figure_ref, Formula_ref,
                      Section_ref, Annexe_ref, Titled_group_ref);
   Affiliation_ref = REFERENCE (Affiliation);
   Annexe =
      BEGIN
         Annexe_title = Lines;
         Sections_seq;
      END;
   Sections_seq = LIST OF (Section);
   Section =
      BEGIN
          Section_title = TEXT;
          Section_body =
             LIST OF ( Paragraphs_or_Relations =
                CASE OF
                   Paragraphs_seq;
                   Relation_def;
                END);
        ? Sections_seq;
      END;
      ...
ASSOC
   Note = Paragraphs_seq;
UNITS
   Subscript = TEXT;
      ...
END
```

3.2 Linguagem P

Devido ao modelo adotado pelo Thot, a apresentação do documento está separada de sua estrutura e conteúdo. O conceito de apresentação engloba o chamado *layout* de página, a composição e o estilo do documento. O esquema de apresentação define o conjunto de operações que exibe o documento na tela ou o imprime. A apresentação do documento é definida por uma linguagem, chemada P.

O elo entre a estrutura e a apresentação [9] é claro: a organização lógica de um documento é usada para compor a apresentação, já que o propósito da apresentação é tornar evidente a organização do documento. O Thot usa uma aproximação de dois níveis, onde a apresentação é inicialmente descrita em termos abstratos, sem levar em consideração cada estrutura em particular, e em seguida a apresentação é realizada sem o achatamento da estrutura dada.

A descrição da apresentação define também a apresentação genérica, já que ela descreve a aparência de uma classe de documentos ou objetos.

Para preservar a homogeneidade entre os documentos, a apresentação é descrita como um simples conjunto de ferramentas que dá suporte suporte ao *layout* de grandes documentos tão bem quanto a composição de objetos, como figuras gráficas ou fórmulas matemáticas.

Para assegurar a homogeneidade das ferramentas para documentos tanto quanto para os

objetos contidos neles, toda apresentação do Thot é baseada na noção de caixa, tal como em Tex. Uma caixa corresponde a cada elemento do documento, pode ser associada a uma string de caracteres, uma linha de texto, uma página, um parágrafo, um título, uma fórmula matemática ou uma célula de uma tabela.

Uma apresentação genérica define os valores dos parâmetros de apresentação, ou uma forma para calcular esses valores para a estrutura genérica. A definição dos parâmetros de apresentação é feita com a linguagem P. Um programa feito nessa linguagem é chamado de Esquema de Apresentação. Ele usa a mesma meta-linguagem que foi usada com a linguagem S para produzir o Esquema de Estrutura.

Um Esquema de Apresentação começa com a palavra-chave *PRESENTATION* e termina com a palavra-chave *END*. A palavra *PRESENTATION* é seguida pelo nome da estrutura genérica a qual a apresentação será aplicada. Este nome deve ser o mesmo nome usado no Esquema de Estrutura.

Para evitar ter que especificar para cada tipo de elemento definido no Esquema de Estrutura, valores para cada um dos numerosos parâmetros de apresentação o Esquema de Apresentação permite a definição de um conjunto de regras de *default*. Essa regras se aplicam a todas as caixas de elementos definidos no Esquema de Estrutura e nas caixas de apresentação e *layout* de página definidos no Esquema de Apresentação.

Uma regra de apresentação define parâmetros de apresentação ou funções de apresentação. Essas funções podem ser a criação de uma caixa de apresentação, o estilo de quebra de linha e quebra de página ou copiar de outra caixa.

Para cada caixa e cada perspectiva, todo parâmetro de apresentação é definido somente uma vez ou explicitamente, ou pelas regras de *default*. Em contraste, as funções de apresentação não são obrigatórias e podem aparecer muitas vezes para o mesmo elemento.

Uma característica importante do Esquema de Apresentação é que este pode definir diferentes vistas para um mesmo documento. Desta forma, pode-se apresentar em uma vista apenas os comentários, fórmulas, ou programas presentes no documento, no lugar de apresentar este como um todo. As vistas são definidas no Esquema de Apresentação e funcionam como um filtro que nos apresenta apenas a parte da estrutura do documento que será visualizada em um dado momento.

O trecho abaixo é um trecho da Estrutura de Apresentação do protótipo do sistema HyperPro, e foi escrito utilizando a linguagem P do Thot:

```
{------------------------------------------------------------------------}
{ Document model for Logic programming environment                       }
{ Ali Ed-Dbali / Khalid El Qorchi/Flavia Peligrinelli                    }
{ MAy, 04th, 1999.                                                       }
{------------------------------------------------------------------------}
PRESENTATION HyperPro;
#define Page_Edit_Width 17 cm
#define Page_Edit_Height 27.7 cm
#define Page_Edit_Top_Margin 1 cm
#define Page_Edit_Left_Margin 2 cm
 ...
{------------------------------------------------------------------------}
VIEWS
Text_view, Table_of_contents, Program_view, Comment_view, Assertion_view, Type_view,
Manual_projection_view, Recursive_projection_view, CRI_view, IV_view, Regular_expr_projection_view
#ifdef PAGE
```

```
   , Vue_infos
#endif
   ;
{-------------------------------------------------------------------------}
DEFAULT
   BEGIN
      HorizRef : Enclosed . HRef;
      VertRef : * . Left;
      Width : Enclosing . Width;
        ...

{-------------------------------------------------------------------------}
BOXES
#ifdef PAGE
   Box_Odd_Page_number:
        BEGIN
           Background : White;
           Foreground : Black;
           Fillpattern : nopattern;
           Content : VarPageNumber;
           VertPos : Top = Previous PAGE_BREAK . Bottom + 0.3 cm;
           HorizPos : Right = Previous PAGE_BREAK . Right;
           Height : 2 cm;
           Size : 11 pt;
           Font : times;
           Style : Roman;
        END;

   Box_Even_Page_number:
        BEGIN
           Background : White;
           Foreground : Black;
            ...
        END;
{-------------------------------------------------------------------------}
RULES
   HyperPro:
        BEGIN
           Justify : No;
           Size : 12 pt;
           Adjust : Left;
#ifdef PAGE
           Width : Enclosing . Width;
           HorizPos : VMiddle = Enclosing . VMiddle;
           Page(Box_Page_definition);
           IN Table_of_contents
              Page(Box_Page_TOC_definition);
           IN CRI_view                          {*CRI*}
              Page(Box_Page_CRI_definition);    {*CRI*}
           IN IV_view                           {*IV*}
               ...
END
```

3.3 Linguagem T

Devido ao seu modelo de documento, Thot pode produzir documentos de forma abstrata em alto nível. Essa forma é chamada de forma canônica e é específica do Thot. Ela se adapta bem às manipulações do editor, mas não se adapta necessariamente a outras operações que possam ser aplicadas aos documentos. Por este motivo, o editor Thot oferece a escolha de salvar documentos na forma canônica ou em um formato definido pelo usuário. Neste caso, o documento Thot é transformado pelo programa de tradução [9]. Essa facilidade pode também ser usada para exportar documentos do Thot para outros sistemas usando outros formalismos.

A tradução pode ser usada para exportar documentos para formatadores básicos como Tex, Latex, Scribe e troff. Também pode ser usado para traduzir documentos para SMGL e HTML.

Para cada documento ou classe de objetos, um conjunto de regras de tradução pode ser definido, especificando como a forma canônica deve ser transformada. Essas regras de tradução estão agrupadas nos Esquemas de Tradução. A mesma estrutura lógica pode ter diferentes Esquemas de Tradução, cada um definindo regras para um formalismo diferente. Os Esquemas de Tradução são genéricos.

Os Esquemas de Tradução são escritos em uma linguagem chamada T. A gramática da linguagem T é especificada usando meta-linguagem e os esquemas de tradução são escritos usando as mesmas convenções dos Esquemas de Estrutura, linguagem S, e Esquemas de Apresentação, linguagem P.

Um Esquema de Tradução começa com a plavra-chave *TRANSLATION*, seguida do nome da estrutura genérica para a qual está sendo definida e um ponto e vírgula e termina com a palavra-chave *END*. O nome da estrutura genérica deve ser o mesmo do Esquema de Estrutura.

A tradução dos elementos que compõe um documento é feita na ordem induzida pela estrutura de árvore, exceto quando a regra Get [9] é usada para mudar a ordem de tradução. Para cada elemento, o tradutor primeiro aplica as regras específicas para o tipo do elemento antes da tradução do seu conteúdo. Se várias regras são aplicadas a esse elemento, o tradutor as aplica na ordem em que elas aparecem no esquema de tradução.

O trecho a seguir é um trecho de programa usando um Esquema de Tradução escrito na linguagem T do Thot para gerar uma parte do programa de um documento HyperPro:

```
{-------------------------------------------------------------------------}
{ Document model for Logic programming environment                        }
{ Pierre Deransart / Ali Ed-Dbali / Khalid El Qorchi                       }
{ January 25th, 1998.                                                     }
{-------------------------------------------------------------------------}
{Thot->LaTeX2e Translation}
TRANSLATION HyperPro;
{-------------------------------------------------------------------------------}
RULES
   HyperPro:
        BEGIN
           Use ParagraphT for Paragraphe;
           Use RefBibT for RefBib;
           Create '%HyperPro to Latex2e - 10 february 1998\12';
           Create '%Document generated automatically (Do not edit !)\12\12';
           Create '%\renewcommand{\psfig}[1]{\mbox{#1}}\12\12'; {sans appel a fig}
           Create '%\title{}\author{}\date{\empty}\12\12';
```

```
            Create '\begin{document}\12\12';
            Create '\newpage\12\tableofcontents\12\end{document}\12' After;
               ...
         END;
   Document_title:
         BEGIN
            Create '\title{' Before;
            Create '}\12' After;
            Get Document_date After;
            Get Authors After;
            Create '\12\maketitle\12' After;
        END;
   Document_date:
         BEGIN
            Create '\12\date{';
            Create '}\12' After;
         END;
             ...
END
```

3.4 Linguagem A

No sistema Thot, a geração da aplicação baseia-se em Esquemas de Aplicação, que são escritos em uma linguagem chamada A [12] (*Generation Application Language*). Esta linguagem é usada para a definição da interface gráfica e para a criação de menus, que podem ser associados às funções padrões do sistema ou a novas funções específicas.

Uma aplicação é formada por comandos e ações. Os comandos são executados ao se escolher um item de menu e as ações são executadas quando os eventos aos quais elas estão associadas ocorrem.

Os eventos foram agrupados de acordo com os objetos aos quais eles são transmitidos: atributos, elementos, regras de aplicação específicas, documentos, visão e aplicação.

Um Esquema de Aplicação se relaciona a um Esquema de Estrutura e possui o mesmo nome deste com a extensão *.a*, ou então é o esquema principal com o nome *EDITOR.a*, onde estão definidos os menus e comandos específicos associados.

Um Esquema de Aplicação começa com a palavra-chave *APPLICATION*, seguida da palavra *EDITOR* ou do nome do Esquema de Estrutura relacionado e termina com a palavra-chave *END*.

Um Esquema de Aplicação pode ser composto de uma ou mais das seguintes seções: *DEFAULT*, que possui associação de eventos e ações e em geral se aplica a todos os tipos de elementos e a todos os atributos definidos no Esquema de Estrutura correspondente; *ELEMENTS*, que contém ações que são chamadas por um elemento; e *ATTRIBUTES*, que define ações que são chamadas por um dado atributo. Pelo menos uma dessas seções deve estar presente nos Esquemas de Aplicação associados.

O Esquema de Aplicação principal, o *EDITOR.a*, possui ainda duas outras seções que não aparecem nos outros esquemas: *USES*, seção opcional que possui os nomes dos outros Esquemas de Aplicação bem como a lista com todos os módulos necessários à aplicação; e *MENUS*, última seção do *EDITOR.a*, que define os menus da barra de menus. Para cada item do menu, o Esquema de Aplicação associa um comando específico exceto para os menus padrão.

O conjunto de ferramentas do Thot permite a declaração de menus em cascata de um nível e ainda suporta a opção de vários idiomas, *multilingual dialogue*. Além dessas facilidades, o kit de ferramentas permite que o usário execute comandos via teclado.

O trecho abaixo é um trecho do Esquema de Aplicação *EDITOR.a* do protótipo do HyperPro Básico:

```
APPLICATION EDITOR;
USES
        { HyperPro schema }
        HyperPro,
        { Index resources }
        Index,
        { Drawing resources }
        Palette,Draw3,
        { Edition mode resources }
        StructEditing,StructSearch,Lookup,StructSelect,DisplayEmptyBox,Keyboards,
        { Tables resources }
        Tableau;
DEFAULT
  BEGIN
   Init.Post -> HP_Event_InitHyperPro;
   ViewOpen.Post -> HP_Event_AfterOpenView;
   DocClose.Pre -> HP_Event_BeforeCloseDoc;
   ElemSelect.Post -> HP_Event_AfterSelEl;
   ElemNew.Post -> HP_Event_DefaultNewElement;
  END;
MENUS
   Main Window:
        BEGIN
        File button:BNew ->                     TtcCreateDocument;
        File button:BOpenDoc ->                 TtcOpenDocument;
        File button:BExit ->                    TtcQuit;
           ...
        END;
  Document Windows:
        BEGIN
        DocFile button:BSave ->                 TtcSaveDocument;
        DocFile button:BSaveAs ->               TtcSaveDocumentAs;
           ...
        Edit button:BInsert ->                  TtcInsert;
        Edit button:BCopy ->                    TtcCopySelection;
           ...
        Present button:BDocument ->             TtcChangePresentation;
        Present.Views button:BVisibility ->     TtcSetVisibilityView;
           ...
        Page button:BPaginateDoc ->             HP_Menu_LocPaginateDocument;
        Page button:BPaginateView ->            HP_Menu_LocPaginateview;
        Page separator;
        Page button:BInsertPB ->                HP_Menu_LocInsertpagebreak;
        Page separator;
        Page button:BPageNumber ->              HP_Menu_LocGotopage;
           ...
        END;
```

```
            ...
END
```

3.5 A API

O conjunto de ferramentas do Thot é um conjunto de funções C que lidam com documentos estruturados no ambiente UNIX / X Windows. Geralmente uma aplicação usa essas funções para, entre outras tarefas, criar documentos novos, modificar documentos existentes, extrair informações de documentos e mostrar partes de documentos.

O conjunto de ferramentas Thot possui cerca de 200 funções agrupadas em grupos onde cada um enfatiza um aspecto diferente do documento, tais como a aplicação, a interface, mensagens, diálogos, documentos; sendo que todas são baseadas no modelo Thot de documentos. Essas funções são acessadas por meio da API (*Application Programming Interface*) [11]. O conjunto de ferramentas do Thot é composto de duas bibliotecas: a biblioteca do *kernel* do Thot e a do editor Thot. A primeira permite que a aplicação lide, de forma automática, com a estrutura lógica e o conteúdo do documento. A segunda contém todas as facilidades incluídas na primeira e ainda acrescenta funções que mostram o aspecto gráfico do documento.

4 HyperPro

O HyperPro [4] [6] [5] é um ambiente experimental que foi projetado para desenvolver programação lógica baseado em programação literária, especificadamente na ferramenta Grif-Thot. Ela oferece uma maneira de lidar com edição de texto e programação CLP (*Constraint Logic Programming* [8]). A edição de texto é feita usando o sistema Thot, que possui facilidade de hypertexto.

O objetivo do HyperPro é documentar programas CLP oferecendo ao usuário a possibilidade de editar, em um ambiente homogêneo e integrado, diferentes programas e versões de programas, comentários, informações de verificação formal, assim como a possibilidade de executar, depurar e testar os programas. Para isso, foi desenvolvida uma estrutura genérica de documentos para programas lógicos de acordo com a metodologia de programação lógica. Os programas são vistos como documentos executáveis. Com poucas alterações, o protótipo do HyperPro pode ser alterado para outras linguagens.

As funções básicas do HyperPro atualmente disponíveis são:

1. diferentes visões do documento;
2. exportações para Latex, ASCII e HTML;
3. índices manuais;
4. tabela de conteúdo;
5. projeções que mostram partes selecionadas de um documento em uma vista separada.

4.1 Estrutura do Documento HyperPro

Um documento HyperPro é basicamente um documento Thot. Ele possui um título, pelo menos uma seção, uma tabela de conteúdo e um índice de referência cruzada.

Em um documento HyperPro, um parágrafo também pode ser uma definição de relação. Uma definição de relação é definida por um título e uma lista de pelo menos uma definição de predicado. O título é um indicador de predicado, nome do predicado e sua aridade, ou um nome.

Já uma definição de predicado é composta por três itens: comentários informais, que são sequências de parágrafos; asserções, que são sequências de linhas de textos opcionais e um conjunto de cláusulas, que podem ser cláusulas Prolog ou CLP.

4.2 Arquitetura do Protótipo

O HyperPro utiliza-se de um editor de texto, o Thot, e sua API. As API's (*Application Program Interface*) do Thot permitem desenvolver aplicações específicas que potencialmente podem atuar na edição de um documento. O kit de ferramentas do Thot é um conjunto de funções de edição, escritas em C, que pode ser usado na construção das API's; tais funções executam operações em ambientes estruturados UNIX / X Window.

O usuário entra com programas escritos na linguagem S e na linguagem P que definem a estrutura genérica do documento. O documento todo é visualizado em uma só vista integral do documento. Além desta vista, mais quatro vistas podem ser definidas. O HyperPro permite definir diferentes esquemas de exportação de documentos na forma canônica para outros tipos de formalismos, como por exemplo, LaTex e ascii.

4.3 Funcionalidades do HyperPro

As principais funções do HyperPro são os índices e as vistas de diferentes partes do documento associados a diferentes utilidades, tais como testes de programas e verificação sintática de linguagens lógicas.

4.3.1 Vistas

O documento do HyperPro pode ser visto por meio de várias perspectivas chamadas vistas, que são especificadas na apresentação genérica. Essas vistas são formas de visualização de partes específicas do programa que são úteis ao programaor durante o estágio do desenvolvimento da aplicação, como por exemplo, a parte dos comentários.

O documento todo é visualizado em uma única vista, denominada vista integral do documento. As vistas podem ser abertas simultaneamente e são automaticamente sincronizadas. Ao invés de escrever na vista integral, o usuário pode editar em uma vista específica, e a informação aparece nas demais vistas abertas. Cinco tipo de vistas foram especificadas, além da vista integral do documento. Essas vistas são:

1. *Comment_View*: é a vista dos comentários, que permite ao usuŕio ver apenas os comentários relativos às definições de predicado;

2. *Assertion_View*: é a vista das asserções, que permite ao usuário visualizar exclusivamente as partes de asserções relativas às definições de predicado;

3. *Typing_View*: é a vista de tipos, que permite ao usuário visualizar apenas os tipos relativos às definições de predicado;

4. *Table_of_Contents*: é a vista do programa que permite que o usuário veja a tabela de conteúdos;

5. *Program_View*: é a vista do programa que permite ao usuário ver somente as partes de *clauses* e definição de predicados.

O usuário pode também especificar a visibilidade de um elemento e também a apresentação de certos elementos que aparecerão em cada vista.

Uma projeção mostra partes selecionadas de um documento em uma vista separada. O processo de seleção depende da projeção desejada. As projeções diferem das vistas no processo de seleção: as vistas já estão incorporadas no Thot, enquanto que as projeções têm que ser implementadas.

4.3.2 Programas e Versões

Um programa é um conjunto de pacotes de cláusulas. Um documento pode conter diferentes programas. O usuário decide como o documento estará organizado, definindo seus programas por meio das seções. O usuário pode definir seus programas selecionando convenientemente no documento suas definições de predicado e colocando referências.

O HyperPro permite que o usuário teste manualmente e automaticamente seu programa.

O usuário pode também definir para qualquer relação de definição, diferentes versões de uma definição de predicado que é documentada e gerenciada com as mesmas utilidades usadas para definir programas. De fato versões de programas são programas que diferem em pelo menos uma cláusula.

4.3.3 Índices

O HyperPro possui os seguintes tipos de índices:

1. Índice de Referência Cruzada: indica onde a relação aparece no documento, onde a sua definição de predicado é encontrada e onde a relação é usada em outros programas ou versões no documento;

2. Índice de Programas e Versões: mostra onde o programa e suas versões foram primeiramente definidos, e está sendo projetado pela equipe francesa.

4.3.4 Conversões e Exportações

O HyperPro é um sistema aberto, o que significa que ele pode trocar documentos com outros sistemas através de um mecanismo de exportação.

O HyperPro produz documentos em um nível abstrato chamado de forma canônica, como foi dito anteriormente. Pode ser definida uma série de regras de tradução de documentos. Na versão atual, foram definidos dois Esquemas de Tradução diferentes: exportação de documentos para LATEX e exportação de documentos para ASCII. Também foram feitos esquemas de exportação de certas partes do documento: exportação do programa, exportação dos comentários informais, exportação das asserções e exportação dos tipos.

5 Desenvolvimento

Durante a fase de estudo do sistema HyperPro foram detectados erros em diversas funcionalidades. A princípio este trabalho visava a correção da funcionalidade de indexação de documentos, mais especificamente o caso do índice de referência cruzada. Este índice não funcionava corretamente para todos os casos. Inicialmente, achamos que o motivo pelo qual o índice não funcionava era porque a paginação no HyperPro não estava implementada. Mas com a implementação desta, que será detalhada na subseção a seguir, não houve mudanças no funcionamento da indexação. O estudo desta também será detalhado mais adiante.

5.1 Paginação de Documentos no HyperPro

Inicialmente a paginação no sistema HyperPro não estava implementada. O Esquema de Aplicação do HyperPro indicava que a função de paginação era acionada via o menu *Page*. Porém essa função não continha nenhum código. Baseando-se no Thot, a função de paginação de documentos foi então implementada.

A paginação no HyperPro foi feita principalmente no seu Esquema de Apresentação, o arquivo *HyperProP.P*, escrito na linguagem P. Como dito anteriormente, o Esquema de Apresentação é baseado na noção de caixa, que pode ser associada à uma página.

As caixas de *layout* de página são criadas implicitamente pelas regras de *layout* de página. Essas regras indicam como o conteúdo de um elemento estruturado deve ser quebrado em linhas e páginas. As caixas de linha e de página não dependem da estrutura lógica do documento, mas sim dos requisitos físicos dos dispositivos de saída, tais como o tamanho do caractere, a altura e a largura da janela na tela ou da folha de papel.

Podem ser definidos Esquemas de Apresentação que não possuem quebras de página. Este tipo de esquema é particularmente interessante na fase inicial de trabalho em um documento, onde a passagem de páginas dificulta a leitura deste mesmo na tela. Nesse caso, os elementos associados, tais como notas, que são normalmente exibidos no rodapé da página são apresentados em uma janela separada. Mas, uma vez que o documento já foi escrito, é desejável que ele seja exibido na tela da mesma maneira que ele será impresso. Desse modo, o esquema de apresentação deve definir páginas.

A linguagem P permite a especificação das dimensões de uma página, assim como a sua composição. É possível também a geração de números de páginas, zonas no pé das páginas para notas, etc. O editor segue esse modelo e insere marcas quebra de página no documento que está sendo usado durante a impressão, assegurando que as páginas no papel sejam as mesmas das telas.

Uma vez que o documento foi editado com um Esquema de Apresentação definindo páginas, ele contém marcas de páginas. Mas também é possível editar um documento sem páginas. Nesse caso, as marcas de página são simplesmente ignoradas pelo editor. Elas só serão novamente consideradas quando o esquema com páginas é usado. Desse modo, o usuário pode escolher entre esquemas com e sem páginas.

Uma quebra de página, no lugar da página em si, é então tratada como uma caixa. Essa caixa de quebra de página contém todos os elementos do rodapé da página, uma regra marcando a margem dessa página, e todos os elementos do cabeçalho da próxima página. Os elementos do rodapé e do cabeçalho podem ser títulos, números de página, elementos associados, por exemplo notas, etc. Todos esses elementos, assim como seu conteúdo e aparência gráfica são definidos pela apresentação genérica.

Um mesmo Esquema de Apresentação então define documentos com e sem paginação. Esse esquema, depois de compilado, irá gerar dois arquivos executáveis: um esquema de documento sem paginação, o arquivo *HyperProP.PRS*, e outro com paginação, o arquivo *HyperProPP.PRS*. Quando a paginação é requerida pelo usuário por meio do menu *Page*, o Esquema de Apresentação com paginação é carregado. Lembramos que o Esquema de Apresentação carregado por *default* é o sem paginação.

No Esquema de Apresentação o tamanho da página é definido, sua altura, sua largura e suas margens. Também a aparência das páginas são definidas nesse esquema, e como o número da página será inserido nela. Há a separação das regras quando o documento será paginado e quando não o será.

Associado ao item de menu relativo a paginação do documento, foi implementada em C a função HP_Menu_LocPaginateDocument. Esta função usa várias funções do kit de API do Thot, e faz a paginação do documento quando o usuário seleciona o item de menu relativo à paginação do documento. Ela funciona do seguinte modo: inicialmente ela pega todas as vistas do documento que estão abertas, e manda paginá-las, chamando a função que pagina vistas. Nessa função as caixas de quebra de página são criadas, assim como os elementos do Esquema de Apresentação que definem uma página, como o rodapé e o número da página.

Associado ao item de menu relativo a paginação de uma vista do documento, foi implementada em C a função HP_Menu_LocPaginateview. Esta função também usa várias funções do kit de API do Thot e faz a paginação da vista do documento que foi selecionada pelo usuário. Ela chama a função que pagina vistas, descrita acima.

Associado ao item de menu relativo a *Go to Page*, foi implementada em C a função HP_Menu_LocGotopage. Esta função foi implementada em C e usa várias funções do kit de API do Thot, e exibe a página pedida pelo usuário. Ela inicialmente constrói a caixa de diálogo que aparecerá para o usuário escolher o número da página que ele quer que seja exibida. A página com essa numeração é então procurada na árvore do documento e exibida.

5.2 Indice de Referência Cruzada

Associado ao item de menu relativo a criação do Índice de Referência Cruzada, foi implementada em C a função HP_Menu_PredCrossRefIndex. Esta função usa várias funções do kit de API do Thot, e constrói o índice de referência cruzada (IRC) quando o usuário seleciona o item de menu relativo à criação do índice. Esta função funciona basicamente em dois passos, descritos a seguir.

No primeiro passo, a função percorre a estrutura do documento HyperPro, para o qual será construído o IRC, acessando e armazenando os elementos necessários para a construção do índice. Então cada relação que participe do documento é acessada, e para cada uma dessas relações, são acessados os elementos correspondentes ao seu título, à sua definição de predicado corrente (c.p.d.) e às chamadas que a relação possa ter no documento. As relações são então organizadas em uma lista e o seu título, seu c.p.d. e suas chamadas são armazenados na sublista da relação correspondente. Essa estrutura é uma lista encadeada de listas encadeadas.

O fato de existir duas relações com mesmo nome e aridade, mas que participem de versões diferentes, possuindo c.p.d. diferente, não afetará o índice. A relação terá duas entradas diferentes no índice, uma para cada versão. Isso é possível, pois cada relação possui um identificador próprio para o elemento no documento HyperPro que a representa.

Para a exibição destas informaçãoe no IRC, foi criado um elemento cujo esquema de estrutura e de apresentação atende às necessidades deste índice.

Já no segundo passo da função para a construção do IRC, as informações coletadas no primeiro passo são utilizadas para a construção do documento correspondente ao IRC, definido pelos esquemas citados acima. Um elemento IRC é criado e automaticamente são criados os elementos correspondentes a cada uma das relações, que correspondem às entradas do índice. Para cada relação, as referências correspondentes ao título, c.p.d. e chamadas à relação são criados e definidos para os elementos correspondentes no documento HyperPro que foram coletados no passo anterior. Essas referências aparecem como o número da página em que aparece os elementos referenciados. Clicando-se nesse número, o elemento é então exibido no documento HyperPro, em sua vista integral.

O que estava acontecendo na execução desta função para construção de irc era o seguinte: o índice era construído, porém como o documento não estava divido em páginas, porque a função de paginação não estava implementada, todas as referências às páginas do título, do c.p.d. e das chamadas à relação eram a página 1, e esta referência não funcionava, selecionando-se esse número o elemento referenciado não era exibido, pois o documento era apenas uma única página muito grande. Além disso, nem as referências às chamadas de certas relações não apareciam, aparecendo apenas um ponto de interregoção no lugar do número da página.

Visto isso, a paginação deveria ser implementada para o correto funcionamento do Índice de Referência Cruzada, e isso foi feito, a paginação de um documento HyperPro foi implementada.

Com o correto funcionamento da paginação estava ocorrendo o seguinte: no índice de referência cruzada, as referências ao título, ao c.p.d. e as chamadas à algumas relações funcionaram corretamente, sendo exibido o número da página onde esses elementos apareciam, e clicando-se nesse número, o elemento era exibido no documento HyperPro. Porém chamadas à algumas relações não estavam funcionamento corretamente, continuava aparecendo um ponto de interrogação no lugar de aparecer o número da página onde a relação era chamada. O código da função de IRC então teve que ser estudado mais a fundo.

Os pontos de interrogação aparecem no lugar das chamadas no caso de relações que não são chamadas dentro da definição de outras relações, como por exemplo as relações principais, que começam o programa. Neste caso, no lugar do ponto de interregoção aparecerá a página da definição da relação. Apareceram também pontos de interrogação em chamadas à algumas outras funções, que não descobrimos similaridades entre elas. Com o estudo do código da função de IRC, descobrimos que essas chamadas não foram armazenadas na lista encadeada que armazena os elementos do IRC.

6 Exemplo

6.1 Paginação

6.1.1 Antes da paginação

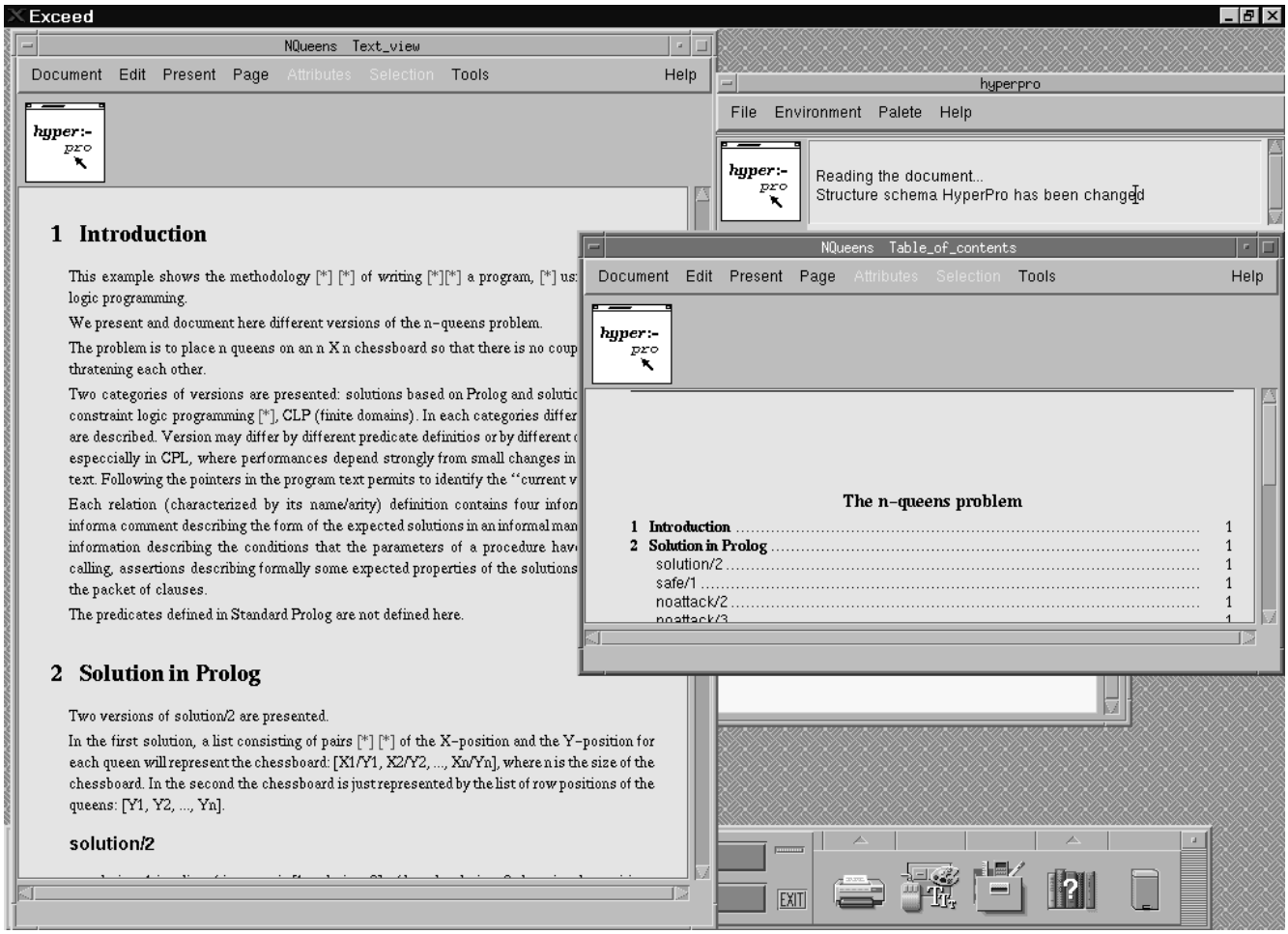

Figura 1: Documento antes de ser dividido em páginas

6.1.2 Depois da paginação

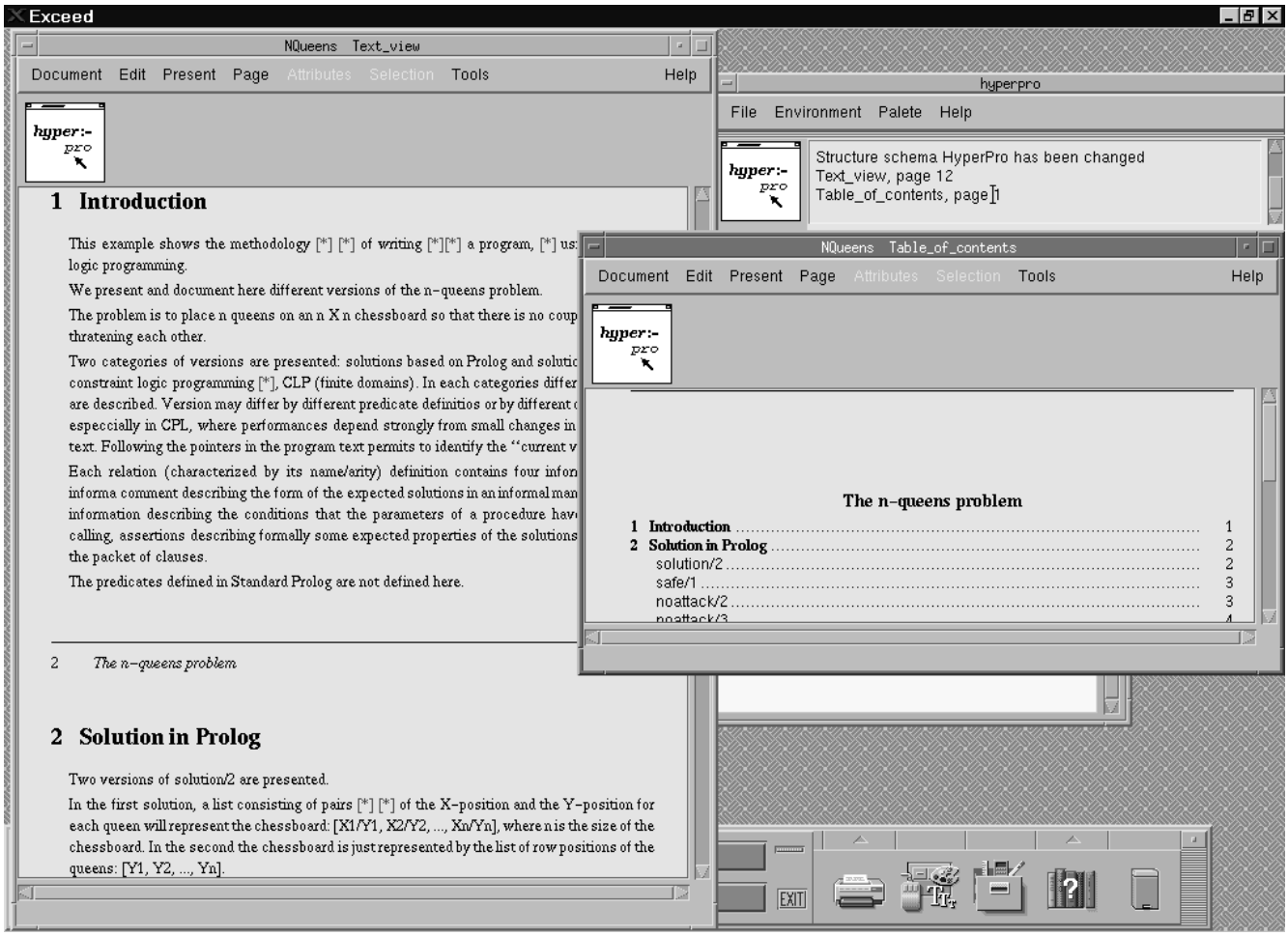

Figura 2: Documento após ser dividido em páginas

6.2 Indice de Referência Cruzada

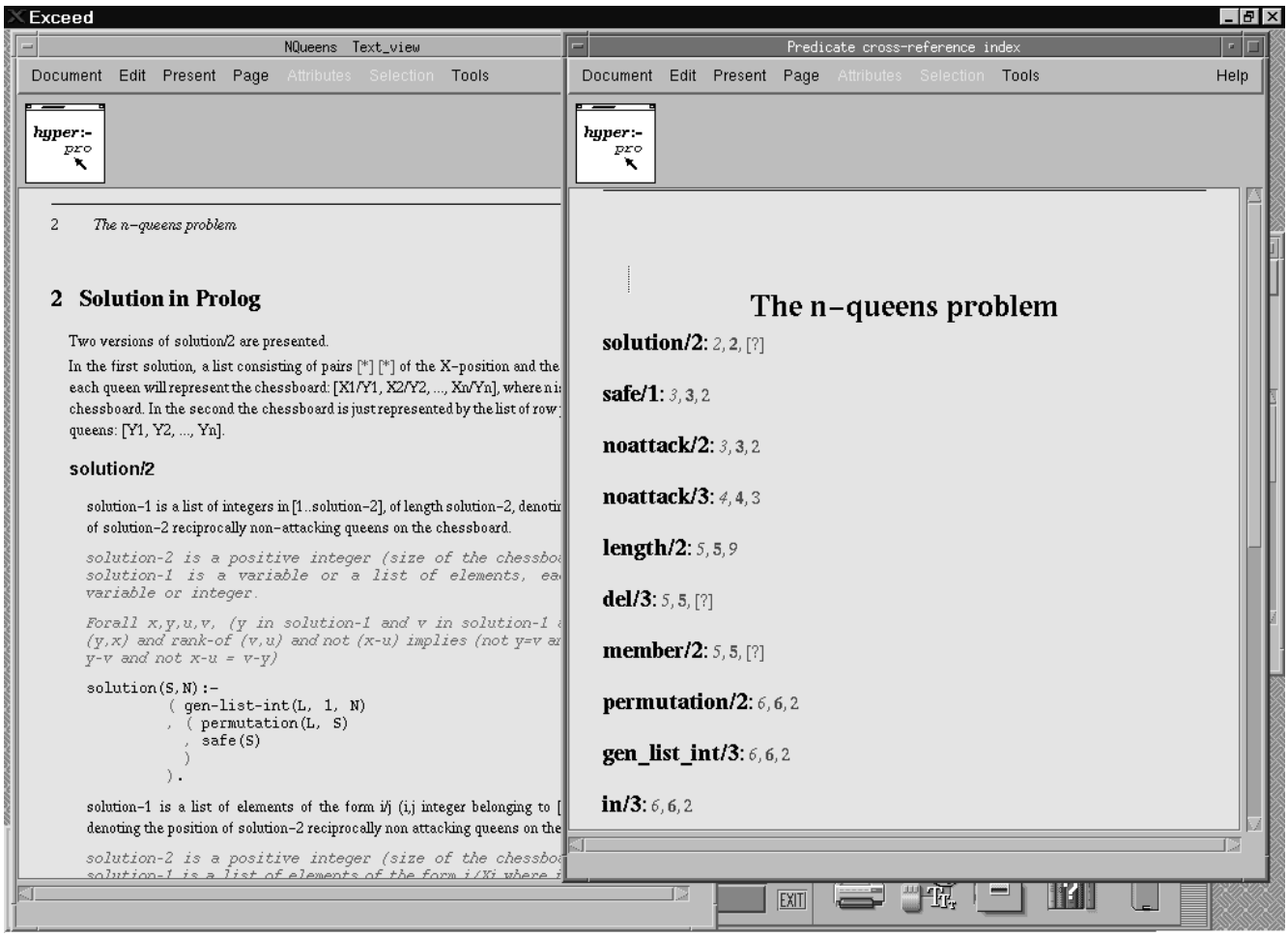

Figura 3: Vista do índice de referência cruzada do documento

7 Conclusão

Este trabalho se propôs a estudar o protótipo do sistema HyperPro, detectar e remover seus erros, garantindo dessa forma, sua robustez para a próxima versão. Foram detectados erros em várias funcionalidades do sistema, mas este trabalho se concentrou, dando prioridade, nas funcionalidades de paginação e indexação de documentos.

A funcionalidade de paginação de documentos já foi implementada e funciona corretamente. As funcionalidades de paginação de vistas do documento e de *Go to page* também foram implementadas e se encontram em funcionamento, como foi dito na seção de desenvolvimento. Já a funcionalidade de índice de referência cruzada foi estudada, e funciona corretamente para algumas relações. Seu código foi estudado e o erro foi descoberto, porém ainda não foi consertado, como dito também na seção anterior.

Dessa forma, o sistema HyperPro Versão 1.1 está mais robusto, tendo funcionalidades importantes que ainda não haviam sido implementadas.

8 Referências Bibliográficas

Referências

[1] Mariza A. S. Bigonha, José de Siqueira, Roberto da S. Bigonha, AbdelAli Ed-Dbali, Pierre Deransart, Fabrício M. Schmidt, and Flávia Peligrinelli. Functionalities and utilities of the hyperpro system. (LLP010/2000), 2000.

[2] Ivan Bratko. *PROLOG Programming for Artificial Intelligence.* Addison-Wesley, 1986.

[3] Nahur Moraes da Fonseca and Mariza A. S. Bigonha. Um ambiente para desenvolver programação em lógica baseado no paradigma de estilo literário. Relatório de Iniciação Científica.

[4] P. Deransart, Roberto S. Bigonha, AbdelAli Ed-Dbali, José Siqueira, and Mariza A. S. Bigonha. A hypertext based environment to write literate logic programms. Relatório Técnico 15, Departamento de Ciência da Computação, ICEx, UFMG, 1996.

[5] P. Deransart, Roberto S. Bigonha, P. Parot, Mariza A. S. Bigonha, and José Siqueira. A hypertext based environment to write literate logic programms. In *Anais do JICSLP'96*, pages 247–252, setembro 1996.

[6] P. Deransart, Roberto S. Bigonha, P. Parot, Mariza A. S. Bigonha, and José Siqueira. A literate logic programming system. In *Anais do I Simpósio Brasileiro de Linguagens de Programação da SBC*, pages 1–16, 1996.

[7] Donald D. Knuth. Literate programming. *The Computer Journal*, 27(2), 1984.

[8] William Leler. *Constraint Programming Language: their specification and generation.*

[9] Vincent Quint. *The Languages of Thot*, 1997.

[10] Vincent Quint, Hélène Richy, Cécile Roisin, and Irène Vatton. *The Thot User's Manual*, 1996.

[11] Vincent Quint and Irène Vatton. *The Thot Tool Kit API*, 1997.

[12] Vincent Quint and Irène Vatton. *The Thot Applicaton Generation Language*, 1997.

[13] Flávia Peligrinelli Ribeiro and Mariza A. S. Bigonha. Hyperpro: Sistema de programas e documentação em um ambiente de programação baseado no paradigma de estilo literário. Relatório Final: POC2.

[14] Fabrício Maia Schmidt and José de Siqueira. Manual de sistema para Índices e projeções baseadas em Índices para o hyperpro básico. Relatório Técnico.

[15] Fabrício Maia Schmidt and José de Siqueira. *Manual do Usuário para Índices e Projeções Baseadas em Índices para o HyperPro Básico*, 1999.

A Manual do Usuário

A.1 Paginação de um Documento

Para o usuário fazer a paginação de um documento HyperPro, deve-se primeiramente abrir o documento em questão. Antes de se acionar o menu *Page*, a opção *Present* na barra de ferramentas do documento ou de alguma vista do documento deve ser escolhida. A seguir, um submenu será aberto e a opção *Present document* deve ser escolhida. Um outro submenu será aberto e a opção *HyperPro* deve ser escolhida, outro submenu aparecerá e a opção *Two sides* deverá ser escolhida. A tabela de conteúdo mostrará então, não apenas os elementos contidos no documento, mas também a página onde eles estão definidos. Como a paginação ainda não foi feita, todas as páginas na tabela de conteúdo serão a página 1. A escolha dessas opções indicará que o documento poderá ser paginado, ele passará do modo em que não pode ser paginado para o modo em que pode ser dividido em páginas.

Para a paginação do documento em si, o usuário deverá escolher a opção *Page* do menu. Um submenu será aberto. Para se paginar um documento, o usuário deverá escolher a opção *Paginate Document*, e o documento será dividido em páginas e a tabela de conteúdo será atualizada, mostrando as páginas onde os elementos estão definidos. Para se paginar uma vista que está aberta, o usuário deverá escolher a opção *Paginate View* no submenu e deverá clicar na vista que ele quer que seja dividida em páginas. A paginação então será efetuada. Se a opção *Go To Page* no submenu for escolhida, uma caixinha de diálogo aparecerá na tela e o usuário deve digitar a página do documento que ele quer que seja exibida, e clicar em *Confirm*. A página escolhida será então exibida. Já a opção *Insert Page Break* foi implementada porém ainda não funciona corretamente.

A.2 Criação do Índice de Referência Cruzada

Para o usuário abrir a vista do índice de referência cruzada de um documento HyperPro Básico, deve-se primeiramente abrir o documento em questão e selecionar na barra de ferramentas do documento, ou de alguma vista do documento, a opção *Tools*. Um submenu será então aberto, e o usuário deverá escolher a opção *Make indexes*. Um outro submenu será então aberto, e o usuário deverá escolher a opção *Predicate cross-reference index*.

A vista do índice de referência cruzada do documento será então aberta.

Aluna: Luciana Leal Ambrosio

Orientadora: Mariza A. S. Bigonha

Co-orientadores: José Siqueira e Flávia Peligrinelli Ribeiro